Process Book

Data Visualization - Prof. Kirell Benzi

Liamarcia Bifano, Nemanja Stojoski,
Milena Filipović

PROJECT FOR DATA VISUALIZATION (COM-480) COURSE @ EPFL

HTTPS://GITHUB.COM/MILENAFILIPOVIC/DATA-VIZ

First release, December 2017

Contents

1 A Brief Introduction 5
1.1 Overview 5
1.2 Motivation 5
1.3 Target Audience 6

2 Discovering what to do... 7
2.1 Related work and inspiration 7
2.2 Our goals 8
2.3 About the dataset 9

3 Our journey 11
3.1 Exploratory data analysis 11
3.2 Possible visualization techniques 11
3.3 Deviations from the initial proposition 11

4 Implementation 13
4.1 Code Architecture 13
4.2 Design and interaction elements 14

5 Evaluation 19
5.1 Lessons learned 19

5.2 Does it work? 19
5.3 Further improvement 19

6 Peer Assessment 21
6.1 Liamarcia Bifano: 21
6.2 Nemanja Stojoski: 21
6.3 Milena Filipović: 22

1. A Brief Introduction

1.1 Overview

The goal of this project is to analyze the behavior of GitHub users, explore the relations among overall programming language usage and the overlaps in the different languages that programmers use. We also wanted to investigate whether changes in usage occur over time.

This project is part of the Data visualization (COM-480) course at the École polytechnique fédérale de Lausanne. The goal of this project is to expand students' data science abilities, to educate them so that they are able to design, judge, build and present their own interactive data visualizations, enabling them to present complex data interactively and effectively.

Upon receiving this assignment we carefully considered a number of potential options that were of great personal interest to us, including the San Francisco bicycle `San Francisco bicycle sharing dataset`, `FBI hate crime dataset` and the `historical flight delay dataset`. All of these would provide us with fertile ground on which we could create our visualizations. However, after much deliberation, we ultimately converged on the same theme - analyzing the behavior of GitHub's users.

This topic was chosen as GitHub itself is something that we use on a daily basis, for a wide variety of different projects, both personal and professional. We decided to take this opportunity to explore the different connections that exist among overall programming language usage, and the overlaps in programmers' language knowledge, through the different languages that they use simultaneously. We also wanted to show whether and how these parameters change over time, taking into account monthly activity from July of 2017 to November of 2017.

1.2 Motivation

Since the first programming language was composed by Ada Lovelace in 1843 [ref history], many important technological revolutions have reshaped the way developers compose their programs. More recently, in the past 45 years almost all of the most popular programming languages had their birth and expansion.

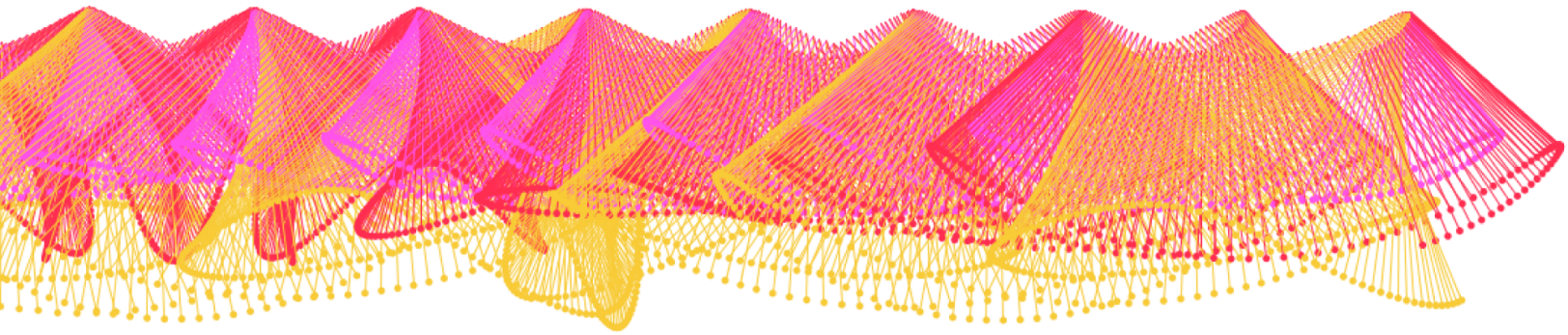
GitHub is a Git version control repository hosting service founded in 2008, with the idea to enable people all over the planet to “build software better, together”. Today, with over 74 million projects by 26 million users in 200 countries who are working in 337 different programming languages, the company represents a unique living and breathing embodiment of what is relevant in the field of technology and computer science. Our team was highly motivated to investigate all of the information that can be extracted from it, in order to gain a different perspective of some of the tools and languages we use every day in our projects and reports. There was no better place to go if one wanted to analyze the ways contemporary programming languages gain and lose popularity across the world. The fact that most of the employees from significant tech companies rely on GitHub for sharing their open source software with the world only encouraged our belief that it symbolizes the main source of all the information about coding and computer science. Ability of GitHub to collect so much of the important data about information technologies made it an obvious (and only) choice for the exploration of evolution and spreading of different programming languages. Additionally, each year the company puts out a report that lays out some of the tech trends, including the most popular programming languages among users of its site.

As Benjamin Whorf formulated in 20th century: "Language shapes the way we think and determines what we can think about.” This stays true in the context of programming languages, as the concepts of chosen programming languages greatly affect the way the final application will look and function.

Like so many members of the tech world, we use GitHub every day but we don’t know how other people use it and what is the behavior of users that are programming in different languages and the relations between languages themselves. We wondered do programmers use several languages at the same time and do they "migrate" to new languages when they appear.

1.3 Target Audience

Our target audience are both GitHub’s users and newcomers to the programming world, who would like to have an overview of the general behavior on GitHub over a specific period of time. Whoever feels like exploring the historical usage of programming languages over the observed period of time should find our visualization quite informative, while at the same time being easy to interpret.
One interesting use case might be dedicated to the developers looking for a programming language to learn next, in order to change the direction of their career or just enrich their knowledge. They can observe all the popular programming languages combinations, through the displayed correlations which will be explained later.



2. Discovering what to do...

2.1 Related work and inspiration

We have encountered several interesting visualizations regarding GitHub's activity and some inspiration for our project included:

- A Parallel coordinates visualization of most used programming languages on GitHub. A screenshot of this can be found on the figure 2.1. Further information can be seen on the following link: link

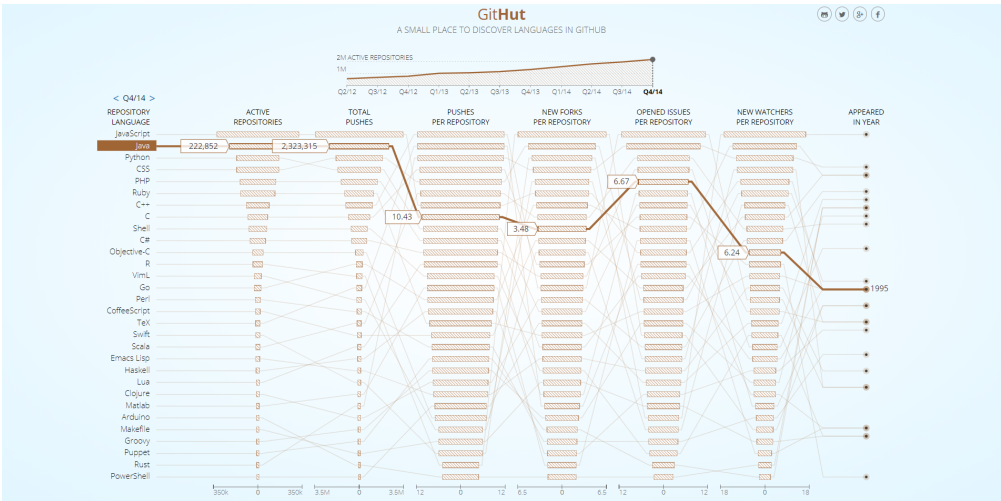


Figure 2.1: GitHut parallel coordinates visualization

- A Jupyter Notebook with visual overview of GitHub event types from 2013. link
- An exploration of expressions of emotions in commit messages. this visualization is both informative and amusing as can be seen on Figure 2.2. link

- A video showing the analysis of millions of GitHub commits - "What makes developers happy, angry, and everything in between?" link

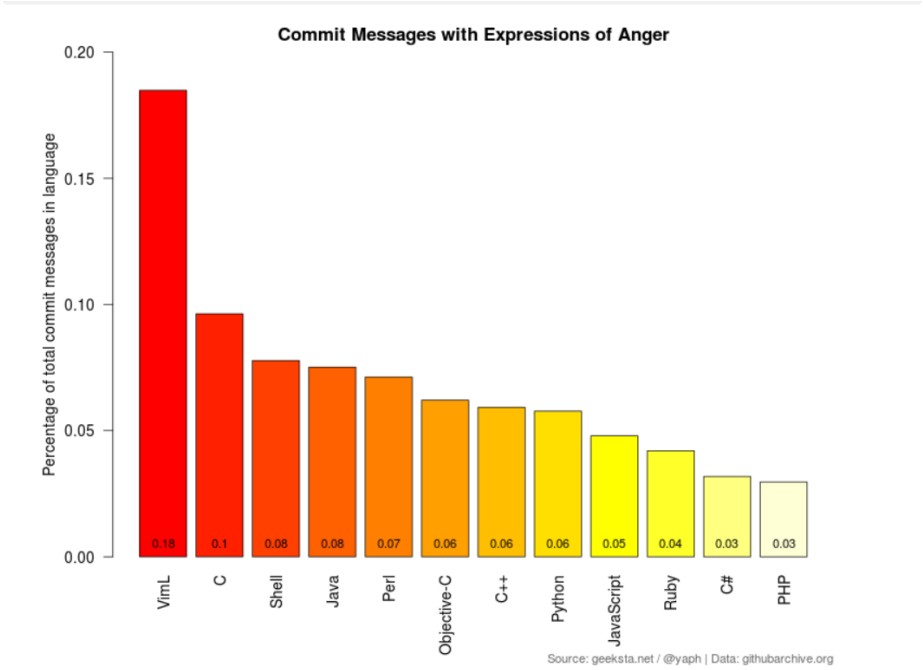


Figure 2.2: Expression of emotions in GitHub commit messages

However none of them explored the interaction of users among different languages.

2.2 Our goals

We wanted to analyze the interaction of users in projects with different programming languages and the number of commits per language and project over time. For instance, one question that can be answered with our visualization can be: "Do users who contribute to projects in Scala also contribute more to projects in Java or Python?" The users are able to compare one programming language to another with the possibility to see which one is the “closest” to their preferred language. Also, our audience is able to see a statistical analysis of the usage of the specific language. By selecting a language, the users will be able to see:

- Average number of commits by Pull Request for the specific language
- Average number actors or people contributing to a specific language
- Average number of changed files by Pull Request
- Average number of commits by Pull Request
- Average number of issues by language
- Average time between open and close a Pull Request

A very broad idea about what we plan to do can be found on the image 2.3 below.

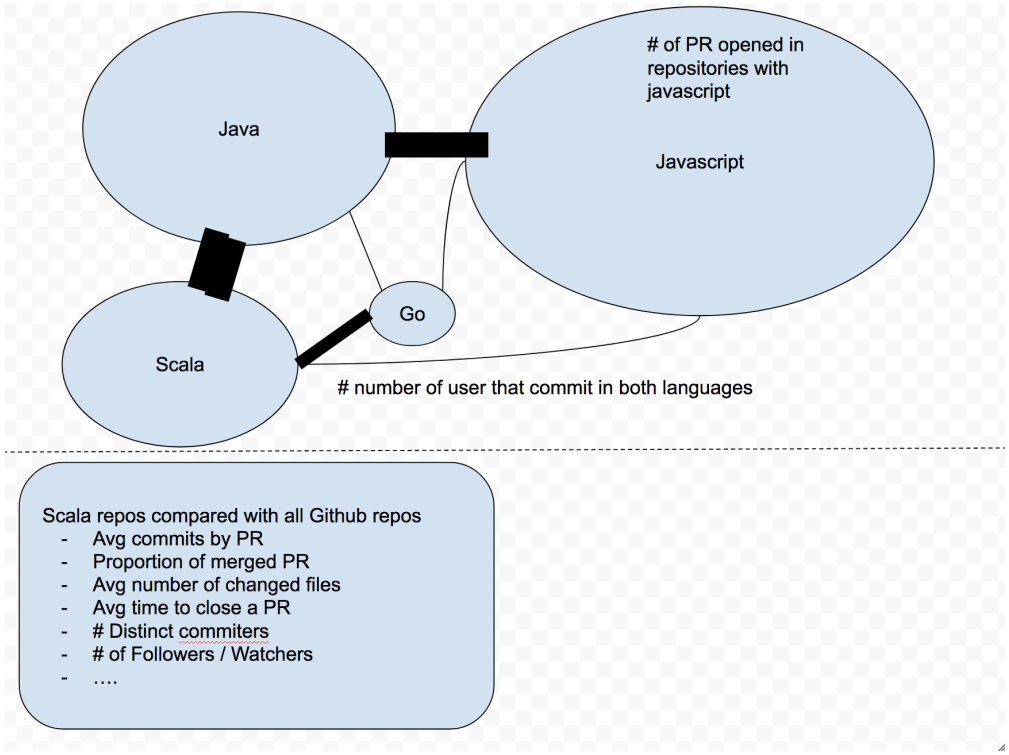
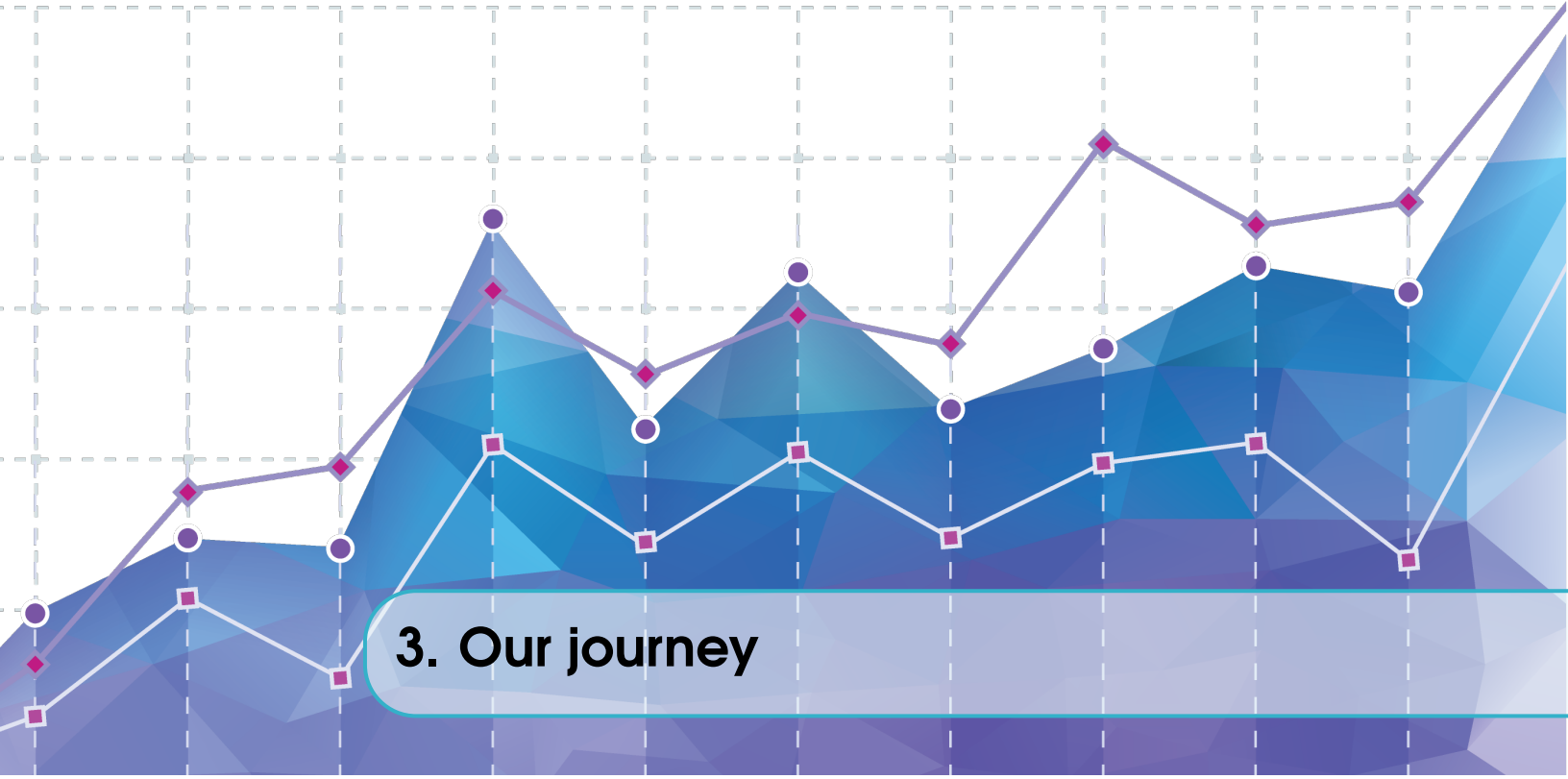


Figure 2.3: A rough initial sketch of our data visualization project

2.3 About the dataset

The dataset that we will used was the GitHub Archive. GitHub Archive is a project with the aim to record the public GitHub timeline, archive it, and make it easily accessible for further analysis. The data is available from 12. February 2011. with the file are split by hour and in a `.json` format.
Another possibility was to download the dataset by querying Google BigQuery however this dataset has a different schema when compared with the .json files, the database is organized by commits and it is not possible filter just Pull Request from the database which means that we would have to deal with a much bigger dataset. Also, BigQuery and it would therefore incur costs on us.
This was the reason we chose to grab the `.json` files ourselves and treat them manually. In order to check all the treatments/filter that is done, you can check here. For each month the script to grab data for a month should run in about 3 hours, but it can vary depending on the internet connection. For this reason, we chose to include only the data from the last 5 months, but our visualization is flexible enough to visualize all the available data, starting from 2011 until the present day. The documentation for the different types of events that are recored in the archive is available on this link.



3. Our journey

3.1 Exploratory data analysis

The first contact with the data was through `python` scripts and initially no visualization was performed, just sanity checks and exploratory analysis to verify the fields described in the documentation exist and are not empty. The `jupyter notebook` we used for that can be found here.
After those checks, we moved on to get the aggregated statistics for each language such as the mean number of actors, Pull Requests, Issues, etc. and the `jupyter notebook` with which we have done that can be found here. Those analyses were performed on a small sample of the data. The reason this was done was because keeping the entire dataset in memory is impossible. This was also the reason that all aggregations on the data were done in `python` beforehand, and not in `javascript` each time that the page is reloaded.

3.2 Possible visualization techniques

Our data is a graph with self-loops where the edges represent the number of unique actors that contributed to two different languages for $i \neq j$ and the number of unique actors when $i = j$. In order to visualize this type of relationship we considered the following layouts of node-link diagrams:

- Multilevel approach
- Collapsible Force
- Orthogonal
- Circle

3.3 Deviations from the initial proposition

At first moment, we considered a standard networked graph such as a `Force-directed` one. However, looking at the data more thoroughly we had realized that there were much more connections among the languages and that a visualization such as that one would be not feasible to render nicely. Besides that, we wanted to include some statistical summary of programming languages so this kind of chart would be very overwhelming.

After exploring some options to visualize this kind of relationships, we decided to move on with Chord Diagram and one of the work that we were inspired by can be seen here. However, our network was drastically unbalanced because there are languages with a huge amount of Pull Requests, which occupy almost the entire chart, leaving very little space for the other ones. To compensate for this, we added the possibility of applying *log* transformation to the connection strength between programming languages and to the arc's size, which represents the number of Pull Requests.
In order to visualize the statistics for each language, we firstly considered a pop-up in the middle of the screen. However, we felt that it would hide all the relationship information and ruin the overall view. At the end, we decided to show detailed information about the specific language by showing a sidebar with different statistics.

4. Implementation

4.1 Code Architecture

The architecture of the code on the website will be described in this section. The only `.html` page is the `index.html`, which of course contains our visualization, as well as the short introduction of the project as well as the instructions for the visualization usage. Most of the d3 code is implemented in the `main.js` JavaScript file, which contains the lines necessary for running the chord diagram part of the visualization. The implementation of the detailed statistics side bar is located in the `metrics-helpers.js` file, and the functions for processing the data are located in `treat-data.js`. Besides this, there are some `.css` files used for styling the visualization, as well as the data itself, which is read and processed by the `.js` files.

4.2 Design and interaction elements

The visualization on Figure 4.1 shows the interdependence between users of multiple programming languages. The amount of space that each language occupies in the chart is proportional to the number of unique contributors and the edges represents the number of common unique contributors. The languages on the diagram are sorted by the programming paradigm they belong to, designated by the different arc colors.

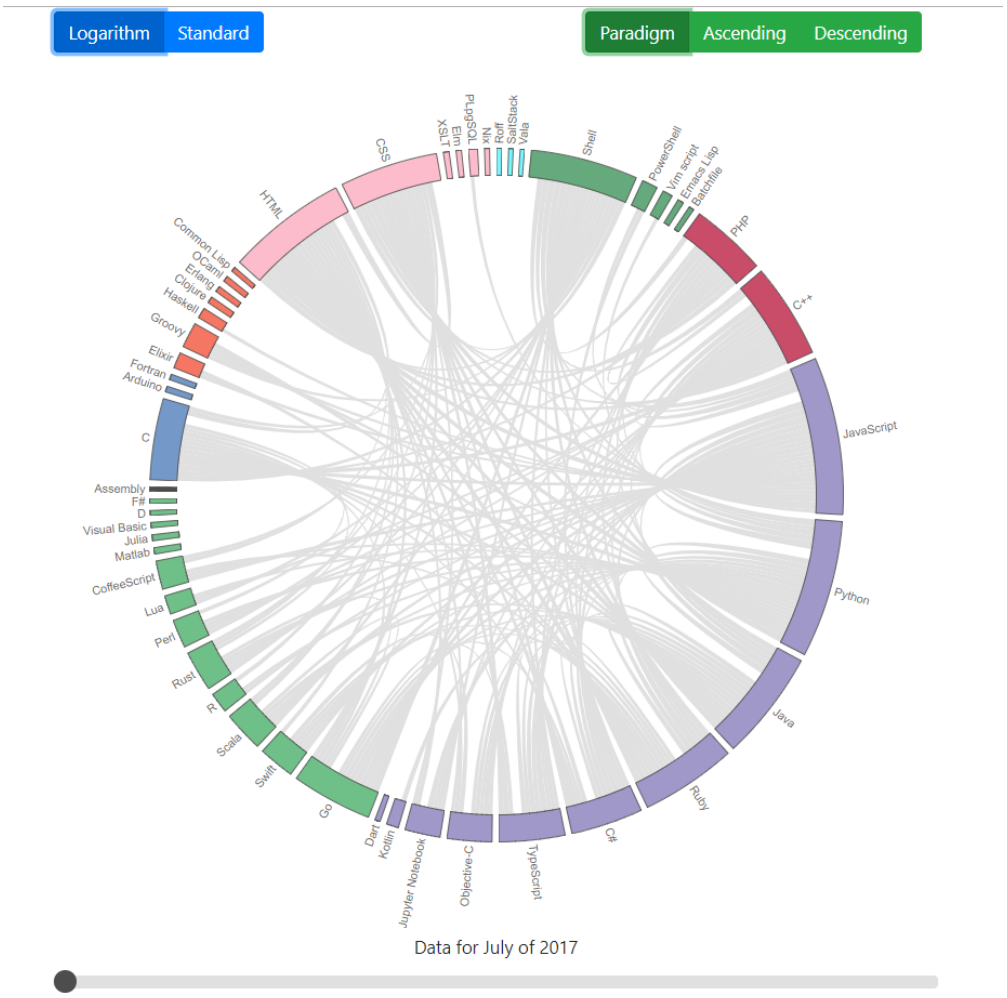


Figure 4.1: Initial status of the language diagram.

To change the sorting, one can do that by clicking on the *Ascending* or *Descending* buttons. Of course, one can also revert to the original sorting, by clicking on the *Paradigm* button. Also, the representation scale can be changed from *Logarithmic* to *Standard*. By dragging the slider below the diagram, the statistics of the language usage will change, to reflect the selected month. This can be seen on Figure 4.2.

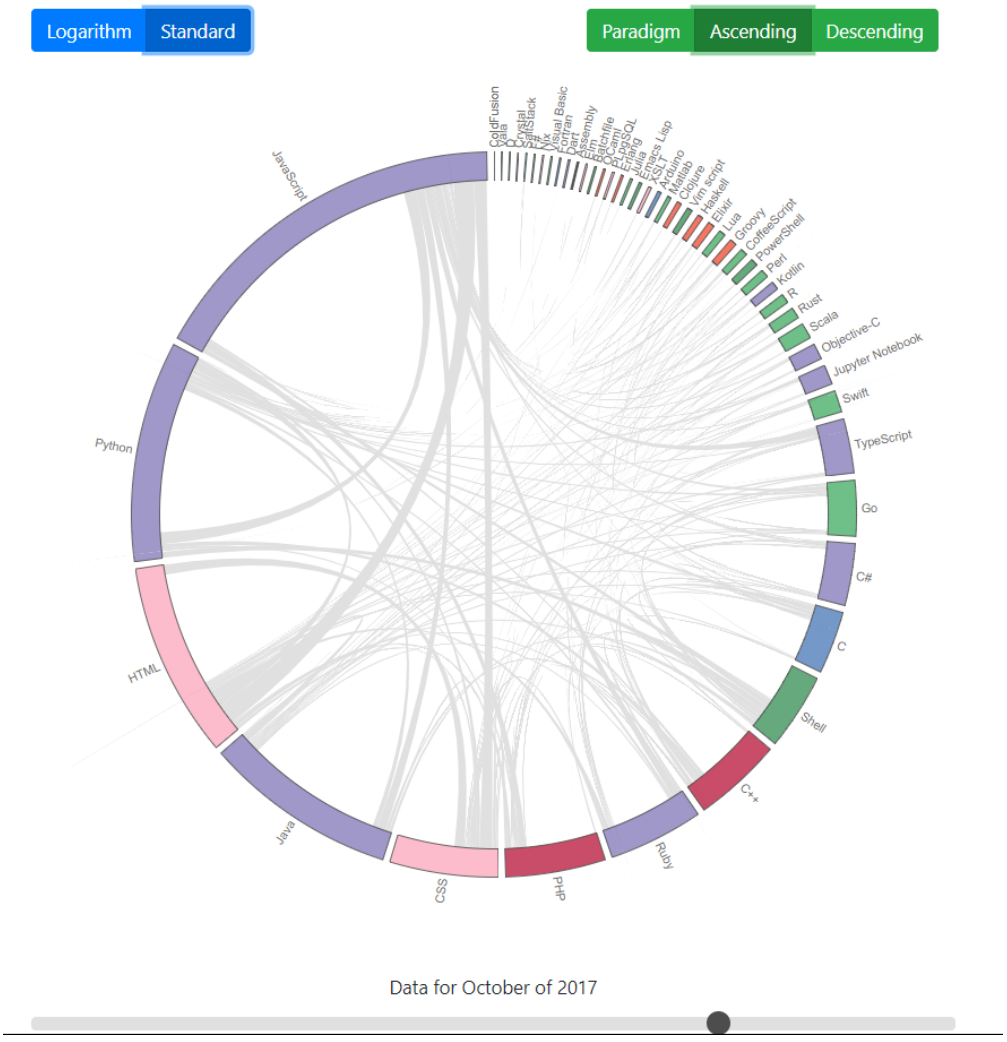


Figure 4.2: The language diagram with modified parameters

By hovering over an arc belonging to a specific language, some general statistics for that language can be seen, such as the average number of Pull Requests per month, average number of actors that used that language per month, as well as the average number of Pull Requests per actor. Also, the connection strength of that language with all the others on the diagram will be shown, which is designated by the thickness of the ribbons that connect the languages. This is shown on the figure 4.3.

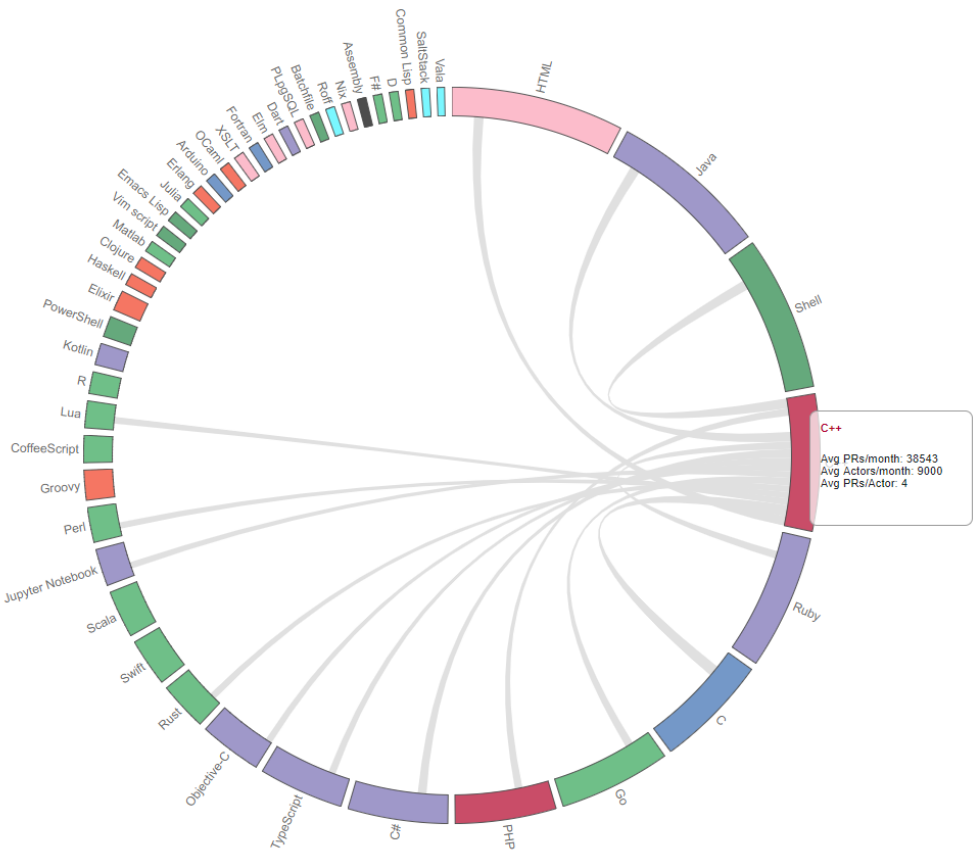


Figure 4.3: Short informations for a language are available by hovering over it.

Additionally, by clicking on the arc belonging to a specific language, more detailed information about the language will be displayed in a separate side bar, including the programming paradigm it belongs to, the languages with which it is correlated the most, and some more statistics about the language itself, and how it compares to the mean value across all languages. These detailed statistics are shown on figure 4.4.

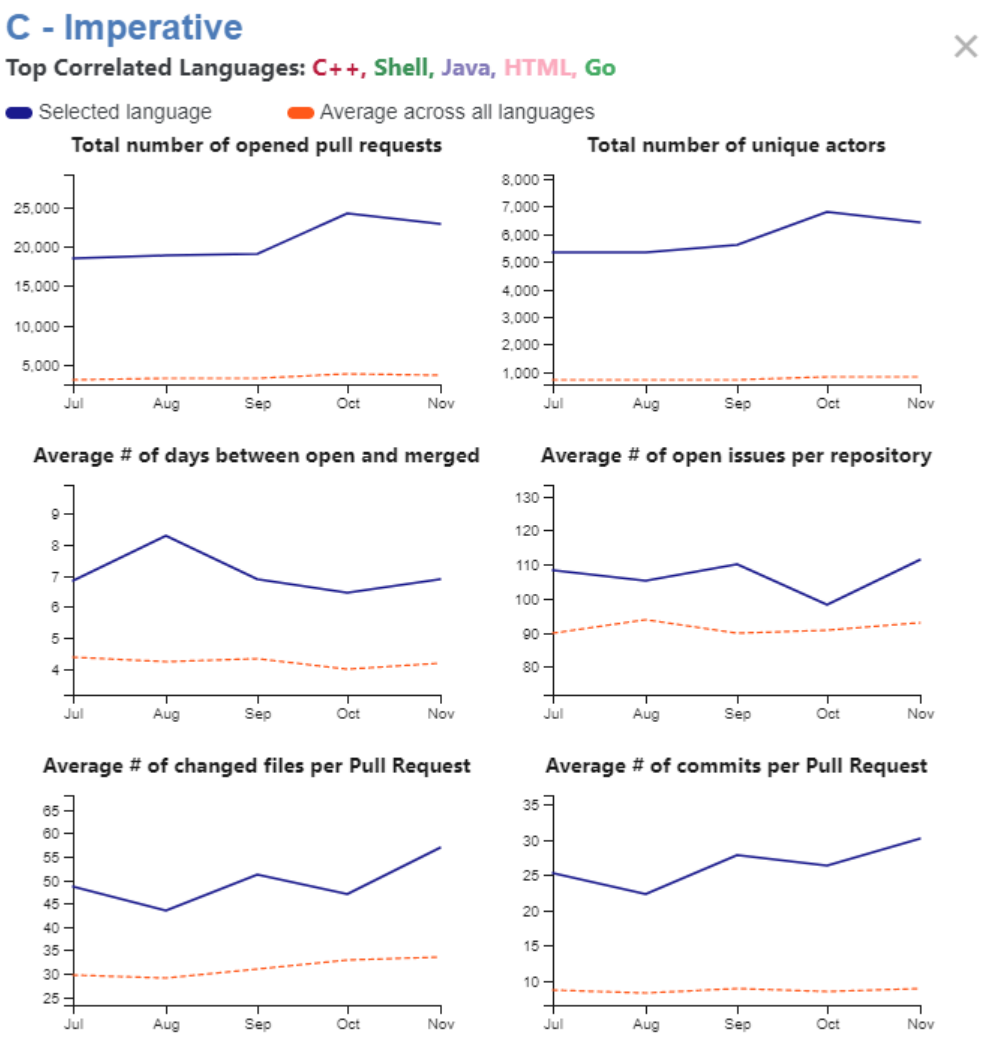


Figure 4.4: Detailed statistics for the chosen language.

If the name of the language is clicked it is removed from the diagram, and can be returned back to the diagram by selecting it in the filter list. An example list of removed languages is shown on figure 4.5.

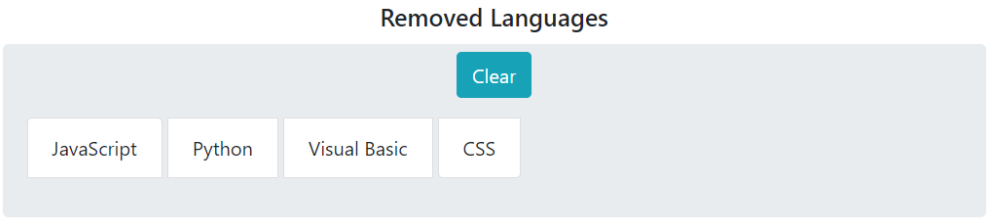


Figure 4.5: A list of removed languages.

5. Evaluation

5.1 Lessons learned

Despite spending around two months looking through our data, while it was represented in a tabular or textual format we found it very hard to draw meaningful conclusions about the connection between different languages. In contrast to this, visualizing our data made it immediately more manageable and easier for the human mind to process. Visualizing this information helped us to understand the message that the data was trying to convey.

5.2 Does it work?

We tried to answer all the questions about our data in an interactive manner. At first glance, our visualization gives an overview of the general relations between programming languages, and allows the viewer to compare the popularity of different programming languages with easy. Further analysis can is also enabled through the changing of the scale used or the way in which the languages are ordered. By hovering over specific arches or clicking on the arches themselves the visualization allows the curious viewer to explore even more. The specific details that are relieved by clicking, can help users understand the specificities of the chosen programming language better.

5.3 Further improvement

The first obvious improvement that can be made on the visualization, without changing its overall look and feel, is to provide language statistics for more months than just five, as is the current number. As stated previously, downloading information from the GitHub Archive is a long, tedious and computationally intensive process. All of the above reasons prevented us from processing more data than we initially hoped we would. Adding more historical data could certainly shed better light on the overall trends of the programmers' language usage, which would provide insightful information for all the users of this visualization.

Another improvement that can be done is to provide even more statistics for each programming language. Depending on the chosen statistics, additional visualization techniques could be

considered.

In the end, there is of course a possibility of making our visualization even more interactive, by generating some information that can be accessed in a three-dimensional, spatial representation.

6. Peer Assessment

6.1 Liamarcia Bifano:

- *Preparation – were they prepared during team meetings?*
  We had different time organization during the semester so the meetings were not so productive as could be if we were synced.
- *Contribution – did they contribute productively to the team discussion and work?*
  Yes,completely.
- *Respect for others' ideas – did they encourage others to contribute their ideas?*
  Yes, we were able to implement all the features that we had in mind even if they had some overlap.
- *Flexibility – were they flexible when disagreements occurred?*
  Yes, we had few disagreements but in all of them everyone was flexible and we were able to work it out.

6.2 Nemanja Stojoski:

- *Preparation – were they prepared during team meetings?*
  Yes, despite having a lot of work to do, they were always prepared and on the point.
- *Contribution – did they contribute productively to the team discussion and work?*
  Absolutely.
- *Respect for others' ideas – did they encourage others to contribute their ideas?*
  The discussions were always productive and we constantly came up with new ideas to implement.
- *Flexibility – were they flexible when disagreements occurred?*
  Disagreements on projects of this scope are always inevitable, but we managed to pull through together and succeed in reaching our goals.

6.3 Milena Filipović:

- *Preparation – were they prepared during team meetings?*
  We did find it hard to meet up during the semester as we all had conflicting and demanding schedules. However, when we did meet up, I feel that those meetings were very productive!
- *Contribution – did they contribute productively to the team discussion and work?*
  Yes! This semester was challenging for al' of us, yet they still found the time to complete the necessary work.
- *Respect for others' ideas – did they encourage others to contribute their ideas?*
  Yes, all ideas were discussed and carefully though about. All members of the team were involved in every decision that was made.
- *Flexibility – were they flexible when disagreements occurred?*
  While there were some minor disagreements, I feel that we overcame them really well and were able to pursue our goals and succeed regardless.